IBM Technical Newsletter

File Number	S360-48
Re: Form No.	C28-6539-4
This Newsletter No.	N28-2226
Date	April 10, 1967
Previous Newsletter Nos.	N28-2214

IBM SYSTEM/360 OPERATING SYSTEM
JOB CONTROL LANGUAGE

This technical newsletter amends the publication IBM System/360 Operating System: Job Control Language, Form C28-6539-4. The attached pages replace pages in the publication. Corrections and additions to the text are noted by vertical bars at the left of the affected text.

Pages to be Inserted	Pages to be Removed
13,14	13,14
19-22	19-22
25,26,26A	25,26
29-32	29-32
35,36	35,36
51,52	51,52
61-64	61-64
67,68	67,68

Summary of Amendments

This technical newsletter includes descriptions of:

- Storage volumes, and the resulting volume states.
- The effect of using the data set name NULLFILE.
- Treatment of channel and unit separation and affinity requests when the automatic volume recognition feature is used.
- New graphic units (Appendix A).

In addition, it clarifies or expands the descriptions of the MSGLEVEL parameter, the operand field of the DD statement, the SPLIT parameter, and the data set sequence number subparameter in the LABEL parameter.

Note: Please file this cover letter at the back of the publication. Cover letters provide a quick reference to changes, and a means of checking receipt of all amendments.

IBM Corporation, Programming Systems Publications, P.O. Box 390, Poughkeepsie, N.Y. 12602

PRINTED IN U.S.A.

JOB STATEMENT

The JOB statement precedes all other statements in the job. It must contain a valid job name in its name field and the word JOB in its operation field. All parameters in its operand field are optional, although your installation can establish that the account number and the programmer's name be mandatory. To follow the flow of parameters in the JOB statement, turn to Appendix E and fold out Chart 1 while reading this chapter.

Identifying the Job (jobname)

The jobname identifies the job to the job scheduler. It must satisfy the positional, length, and content requirements for a name field. No two jobs being handled by a priority scheduler should have the same jobname.

Command statements use certain keywords that you should not use as jobnames:

```
CLOCK       Q
JOBNAMES    T
            A
```

If you find it necessary to use one of these terms as a jobname, you should inform the operator to enclose it in apostrophes if he uses it in a command statement. For example, if you have assigned the jobname CLOCK to a job and the operator wishes to display the status of the job, he must issue a command stating DISPLAY 'CLOCK'. If the apostrophes were omitted, he would get the usual time-of-day display resulting from a DISPLAY CLOCK command.

Supplying Job Accounting Information

For job accounting purposes, the JOB statement can be used to supply information to your installation's accounting procedures. To supply job accounting information, code the positional parameter

```
┌-----------------------------------------┐
|(acct#,additional accounting information)|
└-----------------------------------------┘
```

first in the operand field. Replace the term "acct#" with the account number to which you want the job charged; replace the term "additional accounting information" with other items required by your installation's accounting routines. As a system generation option with sequential schedulers, the account number can be established as a required subparameter. With priority schedulers, the requirement can be established with a cataloged procedure for the input reader. Otherwise, the account number is considered optional.

Notes:

- Subparameters of additional accounting information must be separated by commas.
- The total number of characters in the account number and additional accounting information cannot exceed 142.
- If the list contains only an account number, you need not code the parentheses.
- If the list does not contain an account number, you must indicate its absence by coding a comma preceding the additional accounting information.
- If the account number or any subparameter of additional accounting information contains any special character (except hyphens), you must enclose the number or subparameter in apostrophes (5-8 punch). The apostrophes are not passed as part of the information.

Reference:

- To write an accounting routine that processes job accounting information, see the section "Adding an Accounting Routine to the Control Program" of the publication IBM System/360 Operating System: System Programmer's Guide.

Identifying the Programmer

The person responsible for a job codes his name or identification in the JOB statement, following the job accounting information. This positional parameter is also passed to your installation's routines. As a system generation option with sequential schedulers, the programmer's name can be established as a required parameter. With priority schedulers, the requirement can be established with a cataloged procedure for the input reader. Otherwise, this parameter is considered optional.

Notes:

- The number of characters in the name cannot exceed 20.
- If the name contains special characters other than periods, it must be enclosed in apostrophes. If the special characters include apostrophes, each must be

shown as two consecutive apostrophes, e.g., 'T.O''NEILL'.

- If the job accounting information is not coded, you must indicate its absence by coding a comma preceding the programmer's name.
- If neither job accounting information nor programmer's name is present, you need not code commas to indicate their absence.

Reference:

- To write a routine that processes the programmer's name, see the section "Adding an Accounting Routine to the Control Program" of the publication IBM System/360 Operating System: System Programmer's Guide.

Displaying All Control Statements (MSGLEVEL)

As part of the output for every job, the job scheduler displays the JOB statement, incorrect control statements, and associated diagnostic messages. In addition to this usual output, you can request a display of all the control statements in the job. To receive this additional output, code the keyword parameter

```
MSGLEVEL=1
```

in the operand field of the JOB statement.

If you omit the MSGLEVEL parameter, or code MSGLEVEL=0, only the JOB statement, incorrect control statements, and associated diagnostic messages are displayed.

Note:

- If an error occurs on a control statement that is continued onto one or more cards, only one of the continuation cards is printed with the diagnostics.

Specifying Conditions for Job Termination (COND)

To eliminate unnecessary use of computing time, you might want to base the continuation of a job on the successful completion of one or more of its job steps. At the completion of each job step, the processing program passes a number to the job scheduler as a return code. The COND parameter provides you with the means to test each return code as many as eight times. If any one of the tests is satisfied, subsequent steps are bypassed and the job is terminated.

To specify conditions for job termination, code the keyword parameter

```
COND=((code,operator),..,(code,cperator))
```

in the operand field of the JOB statement. Replace the terms "code" with any decimal number from 0 through through 4095. Replace the terms "operator" with one of the following:

GT	(greater than)
GE	(greater than or equal to)
EQ	(equal to)
LT	(less than)
LE	(less than or equal to)
NE	(not equal to)

If you coded COND=((50,GE),(60,LT)), it would read "If 50 is greater than or equal to a return code, or 60 is less than a return code, I want the remaining job steps bypassed." In other words, the job continues as long as return codes range from 51 through 60.

If you omit the COND parameter, no return code tests are performed.

Note:

- If you want to make only one return code test, you need not code the outer parentheses, e.g., COND=(8,EQ).

Assigning Job Priority (PRTY)
(Pricrity Schedulers Only)

To assign a priority other than the default job priority (as established in the input reader procedure), you must code the keyword parameter

```
PRTY=n
```

in the operand field of the JOB statement. Replace the letter "n" with a decimal number from 0 through 14 (the highest priority number is 14).

If you omit the PRTY parameter, the default jcb priority is assigned to the job.

Requesting a Message Class (MSGCLASS)
(Priority Schedulers Only)

With the quantity and diversity of data in the output stream, your installation may want to separate different types of output

Setting Job Step Time Limits (TIME)
(Priority Schedulers Only)

To limit the computing time used by a single job step or cataloged procedure step, you might want to assign a maximum time for its completion. Such an assignment is useful in a multiprogramming environment where more than one job has access to the computing system.

To assign a time limit to a job step, code the keyword parameter

```
TIME=(minutes,seconds)
```

in the operand field of the EXEC statement. Replace the terms "minutes" and "seconds" with the maximum number of minutes and seconds allotted for execution of the job step. The number of minutes cannot exceed 1439; the number of seconds cannot exceed 59. If the job step is not completed in this time, the entire job is terminated.

If you omit the TIME parameter, the default job step time limit (as established in the cataloged procedure for the input reader) is assumed. If the job step execution time may exceed 1439 minutes (24 hours), code TIME=1440 to eliminate job step timing.

Notes:

- If the time limit is given in minutes only, you need not code the parentheses, e.g., TIME=5.
- If the time limit is given in seconds only, you must code a comma to indicate the absence of minutes, e.g., TIME=(,45).
- When the job step uses a cataloged procedure, you can set a time limit for a single procedure step by including, as part of the keyword TIME, the procedure step name, i.e., TIME.procstepname. This specification overrides the TIME parameter in the named procedure step, if one is present. You can code as many paramters of this form as there are steps in the cataloged procedure.
- To set a time limit for an entire procedure, code the TIME parameter without a procedure step name. This specification overrides all TIME parameters in the procedure, if any are present.

Specifying Main Storage Requirements for a Job Step (REGION)
(Priority Schedulers Only)

For job steps that require an unusual amount of main storage, the EXEC statement provides you with the REGION parameter. Through this parameter you can specify the maximum amount of main storage to be allocated to the associated job step. This size must take into account the system components required by your installation.

To specify a region size, code the keyword parameter

```
REGION=nnnnnK
```

in the operand field of the EXEC statement. Replace the term "nnnnn" with the number of 1024-byte areas you want allocated to the job step, e.g., REGION=51K. This number can range from one to five digits.

If you omit the REGION parameter, the default region size (as established in the cataloged procedure for the input reader) is assumed.

Notes:

- If you have specified a REGION parameter in the JOB statement, REGION parameters in the job's EXEC statements are ignored.
- When the job step uses a cataloged procedure, you can request a region size for a single procedure step by including, as part of the REGION parameter, the procedure step name, i.e., REGION.procstepname. This specification overrides the REGION parameter in the named procedure step, if one is present. You can code as many parameters of this form as there are steps in the cataloged procedure.
- To request a single region size for an entire cataloged procedure, code the REGION parameter without a procedure step name. This specification overrides all REGION parameters in the procedure, if any are present.

Reference:

- The storage requirements you must consider when specifying a region size are outlined in the publication *IBM System/360 Operating System: Storage Estimates*.

DD STATEMENT

Data sets used by processing programs must be represented by DD statements in the input stream. The DD statements pertaining to a particular job step follow the EXEC statement associated with the step. A DD statement must contain the term DD in its operation field. Although all parameters in the DD statement's operand field are optional, a blank operand field is invalid.

The DD statement is the final source of information that is needed to retrieve and store data. Figure 2 illustrates the sources of information and the means by which each source refers to the next.

An input/output macro-instruction, coded as part of the processing program, issues an input or output request (OPEN, CLOSE, GET, PUT, READ, WRITE). This request uses a dcbname to refer to a data control block created earlier by a DCB macro-instruction. The data control block contains information about a data set that is gathered from several sources, one of which is a DD statement whose ddname matches the ddname given in the data control block. The DD statement, the final source of information, is associated with a specific data set. It refers to the data set with a data set name.

Because of the DD statement's position in this sequence of information sources, you can specify such characteristics as buffer size, record length, and device type at the time the job step is executed, rather than when you code the processing program.

To follow the flow of parameters in the DD statement, turn to Appendix E and fold out Chart 2 while reading this chapter. Individual parameters are shown in detail in a series of figures on Chart 3 of Appendix E.

Identifying the DD Statement (ddname)

The ddname identifies the DD statement so that subsequent control statements and the data control block can refer to it. It must satisfy the position, length, and content requirements for a name field. Each ddname within a job step should be unique. If duplicate ddnames exist, all references are directed to the first such DD statement in the job step, and the second is ignored.

Note:

- Omit the ddname if the data set is concatenated with the data set defined by the preceding DD statement, or the DD statement is one of a group of DD statements that define an indexed sequential data set.

If the job step uses a cataloged procedure, the ddname must be qualified by the procedure step name, i.e., procstepname.ddname. The ddname can identify either a DD statement in the procedure, whose parameters you want to override, or a new DD statement you want to add to the procedure. In both cases, the modification is valid only for the duration of the job step; it does not change the procedure permanently.

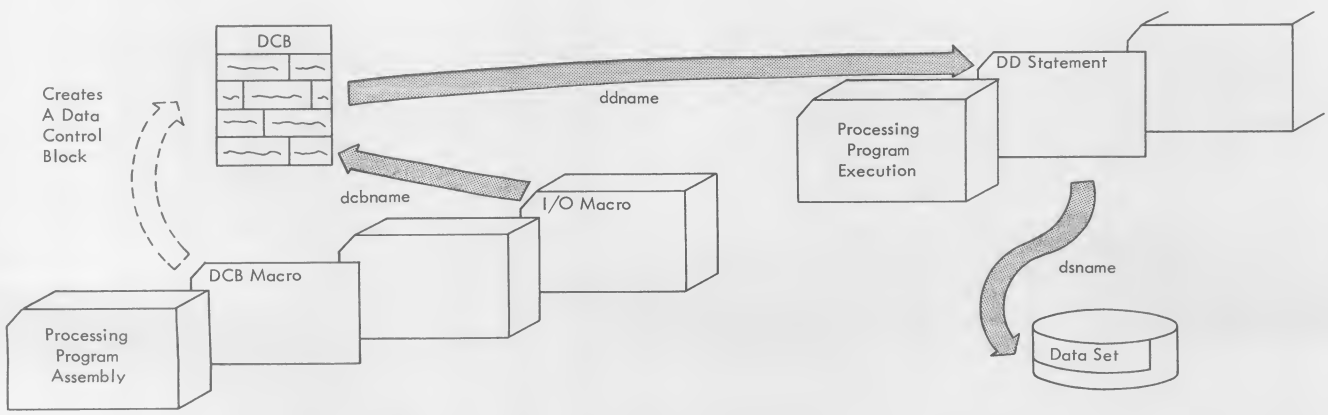

Figure 2. Data Set Information Sources

References:

- For more information on how to code the ddname when creating indexed sequential data sets, see Appendix D of this publication.
- Instructions for concatenating data sets and modifying cataloged procedures are contained in the chapters "Additional DD Statement Facilities" and "Using Cataloged Procedures" in Section 2 of this publication.

Defining Data in an Input Stream (DD * or DD DATA)

One of the ways a data set can be introduced to the system is by its inclusion in an input stream in the form of 80-byte records. A data set in an input stream is bounded by a DD statement that marks its beginning and a delimiter statement that marks its end.

To mark the beginning of the data set, code an asterisk (*) in the operand field of the DD statement that precedes it. If the data contains job control statements (statements with // in columns 1 and 2), as would be the case if the data set were a procedure being cataloged, code the word DATA in the operand field instead of the *. In both cases, do not code any other parameters.

To mark the end of the data set, insert a delimiter statement (/* in columns 1 and 2) after the last data card.

Notes:

- When using a sequential scheduler, you can include only one data set in the input stream for each job step or procedure step. This data set must be defined by the last DD statement in the step. With systems providing multiprogramming with a fixed number of tasks, the processing program that uses data in the input stream must be in the lowest priority partition.
- When using a priority scheduler, you can include more than one data set in the input stream. If you leave out the DD * statement for the first such data set, the system assigns the name SYSIN to it. The delimiter statement is not required following data sets preceded by DD *.
- Data sets in an input stream cannot contain statements having the characters /* in columns 1 and 2.

Bypassing I/O Operations on the Data Set (DUMMY)

The DUMMY parameter, a DD statement positional parameter, offers you the facility to bypass I/O operations, space allocation, and disposition of data sets referred to by the basic or queued sequential access methods. This facility can be used to suppress the writing of certain output data sets, such as assembler listings, and to update new master files with a dummy detail file. Bypassing operations on noncritical data sets also results in a saving of time when you are testing a program. To use this facility, code the word DUMMY as the first parameter in the operand field.

An attempt to read a "dummy" data set in your processing program will result in an immediate end-of-data-set exit. If you attempt to write on a dummy data set, the write request is recognized, but no data is transmitted. In addition, no device allocation, external storage allocation, or data set disposition takes place. Another way of establishing a dummy data set is by assigning it the name NULLFILE.

Postponing the Definition of a Data Set (DDNAME)

A DD statement in a cataloged procedure and, in certain cases, in an input stream, need not contain descriptive parameters. Instead, it can point to a subsequent DD statement that contains a complete description of a data set. The original data set does not assume real characteristics until the DD statement that contains the complete description is encountered. To postpone the definition of a data set in this manner, code

```
DDNAME=ddname
```

in the operand field of the DD statement. Replace the term "ddname" with the name of the DD statement that contains complete information. This feature is particularly useful when a cataloged procedure uses data in an input stream.

Reference:

- For more information and examples of usage of the DDNAME parameter, see the chapter "Using *Cataloged Procedures*" in Section 2 of this publication.

Routing a Data Set Through the Output Stream (SYSOUT)

Output data sets can be routed through the system output stream and handled much the same as system messages. When using a sequential scheduler, you can route a data set through the output stream by coding the keyword parameter

```
SYSOUT=A
```

in the operand field of the DD statement. Your processing program writes the data set on the system output device. A unit record or labeled tape device becomes the system output device when the operator activates it as such with a START WTR command. With systems providing multiprogramming with a fixed number of tasks, the processing program that writes the data must be in the lowest priority partition.

When you use a priority scheduler, such operations are not performed during execution of the job step. Instead, the processing program writes the data set on an intermediate direct-access device. Later the data set is routed through an output stream to a system output device. To schedule such an operation, code the keyword parameter

```
SYSOUT=x
```

in the operand field. Replace the letter "x" with an alphabetic (A-Z) or numeric (0-9) character. The letter represents one of the system output classes. Output writers route data from the output classes to system output devices. The DD statement for this data set can also include a unit specification describing the intermediate direct-access device, and an estimate of the space required. If you omit these parameters, the job scheduler provides default values as the job is read and interpreted.

When using a priority scheduler, you have two additional options with the SYSOUT parameter. If you have a special program to handle output operations, code

```
SYSOUT=(x,progname)
```

Replace the term "progname" with the member name associated with your program. The program must reside in the system library. If you want the data set printed or punched on a specific type of output form, code

```
SYSOUT=(x,,form#)
```

Replace the term "form#" with the 4-digit form number to be used. This form number is used to instruct the operator in a message issued at the time the data set is to be printed.

Note:

- If you wish to specify both an output program and a form number, code SYSOUT=(x,progname,form#).

Identifying the Data Set (DSNAME)

Data sets used by a processing program are identified with the DSNAME parameter. You need not code this parameter if the data set is temporary; the system automatically assigns a name if the DSNAME parameter is omitted. You can specify this parameter in one of three ways:

1. By giving the name by which a data set is or will be cataloged or tabulated.
2. By referring to an earlier DD statement that gives its cataloged or tabulated name.
3. By giving a temporary name, if it is a temporary data set.

To supplement this discussion of the DSNAME parameter pictorially, turn to Appendix E and fold out Chart 3.

1. Data sets that will be identified in later jobs by name, or that were assigned a name in an earlier job are referred to by a cataloged or tabulated name. To name or retrieve a data set of this type, code the keyword parameter

   ```
   DSNAME=dsname
   ```

   in the operand field of the DD statement. Replace the term "dsname" with the data set's cataloged or tabulated name. If the catalog makes use of index levels, you must give a fully qualified name, e.g., A.B.LINKFILE.

   Notes:

   - The data set is assigned a dummy status if you code DSNAME=NULLFILE.
   - If the DD statement refers to a particular generation of a generation data group, you must code the

contain a new data set. DD statements required by operating system utility programs are an exception to this note, in that they require the specification of DEFER.

To optimize the flow of input and output data in a job step, you might want to assign separate access mechanisms to direct-access data sets. To request a separate access mechanism, code the keyword subparameter SEP=(ddname,...,ddname) in the last positions of the UNIT parameter. Replace the terms "ddname" with the names of other DD statements in the job step, e.g., UNIT=(,P,SEP=(INPUT1, INPUT2)). The system will attempt to process this data set with different access mechanisms than the ones used to process the data sets defined by the named DD statements.

Notes:

- When you use a sequential scheduler, the operating system ignores a SEP request if it conflicts with another request, or if sufficient access mechanisms are not available.
- When you use a priority scheduler, the operator is notified if the request cannot be satisfied. He must decide whether to continue the job or to withdraw separation requests.
- Unit separation requests are ignored when the automatic volume recognition feature is used.
- If the list of ddnames includes only one ddname, you need not code the inner parentheses, e.g., UNIT=(190,SEP=INPUT).

2. To conserve the number of units used in a job step, you can request that a data set be assigned the same unit(s) as assigned to an earlier data set in the same step. This technique, known as unit affinity, indicates that you want certain data sets and their associated volumes to use a single unit in sequential order. (Thus, deferred mounting is implied for data sets requesting unit affinity.) To request unit affinity, code the keyword parameter

```
UNIT=AFF=ddname
```

in the operand field. Replace the term "ddname" with the name of an earlier DD statement in the job step. This data set will be assigned the same unit or units as the data set defined by the named DD statement.

Note:

- Unit affinity requests are ignored when the automatic volume recognition feature is used.

Specifying Volume Information (VOLUME)

Information about the volume or volumes on which an input data set resides, or on which an output data set will reside is given in the VOLUME parameter. Volumes can be used most efficiently if you are familiar with the states a volume can assume. Volume states involve two criteria: the type of data set you are defining and the manner in which you request a volume.

Data sets can be classified as one of two types -- temporary or nontemporary. A temporary data set exists only for the duration of the job that creates it. A nontemporary data set can exist after the job is completed. You indicate that a data set is temporary by coding:

- DSNAME=&name.
- No DSNAME parameter.
- DISP=(NEW,DELETE), either explicitly or implied, e.g., DISP=(,DELETE).
- DSNAME=reference, referring to a DD statement that defines a temporary data set.

All other data sets are considered nontemporary. If you attempt to keep or catalog a passed data set that was declared temporary, the system changes the disposition to PASS unless DEFER was specified in the UNIT parameter. Such a data set is deleted at the end of the job.

The manner in which you request a volume can be considered specific or nonspecific. A specific reference is implied whenever you request a volume with a specific serial number. Any one of the following conditions denotes a specific volume reference:

- The data set is cataloged or passed from an earlier job step.
- VOLUME=SER is coded in the DD statement.
- VOLUME=REF is coded in the DD statement, referring to an earlier specific volume reference.

All other types of volume references are nonspecific. (Nonspecific references can be made only for new data sets, in which case the system assigns a suitable volume.)

The state of a volume determines when the volume will be demounted and what kinds of data sets can be assigned to it. The system determines the precise state of a volume by combining two characteristics, the "mountability" of the volume, and its availability for allocation. Some volumes are assigned a permanent state at system generation; others can assume different states, through MOUNT commands and VOLUME parameters in DD statements. The remainder of this discussion on volume states is divided into two parts, the first dealing with direct-access volumes and the second with magnetic tape volumes.

Direct-Access Volumes: Direct-access volumes differ from tape volumes in that they can be shared by two or more data sets processed concurrently by more than one job. Because of this difference, direct-access volumes can assume different volume states than tape volumes. The volume state is determined by one characteristic from each of the following groups:

Mount Characteristics	Allocation Characteristics
Permanently Resident	Public
Reserved	Private
Removable	Storage

All combinations of characteristics are valid except removable/storage. Table 1 explains how direct-access volumes are assigned their mount and allocation characteristics. Actions 4 through 8 are induced by control statements in the input stream.

Permanently resident volumes are always mounted. The permanently resident characteristic applies automatically to:

- All physically nondemountable volumes, such as 2301 Drum Storage.
- The volume from which the system is loaded (the IPL volume).
- The volume containing the system data sets SYS1.LINKLIB, SYS1.PROCLIB, and SYS1.SYSJOBQE.
- Volumes used by the system for SYSIN and SYSOUT. (Priority schedulers only.)

Any other direct-access volume can be designated as permanetly resident in a special member of SYS1.PROCLIB named PRESRES. The reserved characteristic applies to volumes that remain mounted until the operator issues an UNLOAD command. They can be reserved by either a MOUNT command referring to the unit on which they are mounted, or an entry in PRESRES. The removable characteristic applies tc all volumes that are neither permanently resident nor reserved. Removable volumes do not have an allocation characteristic when they are not mounted. A reserved volume becomes removable after an UNLOAD command is issued for the unit on which it resides.

The allocation characteristics -- public, private, and storage -- deal with a volume's availability to be assigned by the system to temporary data sets, and, if the volume is removable, when it is to be demounted. A public volume is used primarily for temporary data sets and, if it is permanently resident, for frequently used data sets. It must be requested by a specific volume reference if a data set is tc be kept or cataloged on it. If a public volume is removable, it is demounted only when it's unit is required by another volume. You can change a removable/public volume to private by specifying VOLUME=PRIVATE. A private volume must be requested by a specific volume reference, and, therefore, cannot contain temporary data sets having nonspecific volume requests. If it is reserved, it remains mounted until the operator issues an UNLOAD command for the unit on which it resides. If it is removable, it will be demounted after it is used unless you specifically requested that it be retained

Table 1. Direct-Access Volume States

Mount Characteristic	Allocation Characteristic		
	Public	Private	Storage
Permanently resident	1 PRESRES entry	2 PRESRES entry	3 PRESRES entry
Reserved	4 PRESRES entry -or- MOUNT command	5 PRESRES entry -cr- MOUNT command	6 PRESRES entry -or- MOUNT command
Removable	7 VOLUME=PRIVATE not specified in the DD statement	8 VOLUME=PRIVATE specified in the DD statement	9 Invalid state

(VOLUME=RETAIN) or passed (DISP=PASS). Once a removable volume has been made private, it will ultimately be demounted. To use it as a public volume, you must have it remounted. A storage volume is always permanently resident or reserved; it is effectively an extension of main storage. Its principal use is to keep or catalog a data set having a non-specific volume reference.

Magnetic Tape Volumes: The volume state of a reel of magnetic tape is also determined by a combination of mount and allocation characteristics:

Mount Characteristics	Allocation Characteristics
Reserved	Private
Removable	Scratch

The reserved/scratch combination is not a valid volume state. Reserved tape volumes assume their state when the operator issues a MOUNT command for the unit on which they reside. They remain mounted until the operator issues a corresponding UNLOAD command. Reserved tapes must be requested by a specific volume reference.

A removable tape volume is assigned the private characteristic when one of the following occurs:

in the operand field of the DD statement. Replace the word "dsname" with the data set's cataloged name. The volume that contains this data set must be mounted before the execution of the job step containing the copy request. A permanently resident volume is the most likely place from which to copy such information, in that it is always mounted.

If such a data set does not exist, you still might be able to copy the DCB parameter of an earlier DD statement in the job. To refer to this DD statement, code the keyword parameter

```
DCB=*.stepname.ddname
```

in the operand field. Replace the terms "stepname" and "ddname" with the job step name and DD statement name, respectively.

Notes:

- If the earlier DD statement is contained in the same job step, you need not code the stepname, i.e., DCB=*.ddname.
- If the earlier DD statement is contained in a cataloged procedure step, you must include the procedure step name, i.e., DCB=*.stepname.procstepname.ddname.

If you wish to modify the information that is copied from another data set label or DCB parameter, code

```
DCB=(reference,list of attributes)
```

Replace the term "reference" with dsname or *.stepname.ddname. The attributes in the list override the corresponding copied attributes. Data set attributes are coded in the form of keyword subparameters separated by commas, e.g., BLKSIZE=810 for a block size of 810 bytes. These subparameters correspond to operands in the DCB macro-instruction and are coded using the same keywords and values. A glossary of valid DCB subparameters is given in Appendix B of this publication.

If you cannot copy another data set label or DCB parameter, you must supply all DCB attributes that are not specified in the processing program (either directly or by default) or data set label. Code the keyword parameter

```
DCB=(list of attributes)
```

in the operand. Again, the attributes are coded as keyword subparameters separated by commas, e.g., DCB=(RECFM=FB,LRECL=80,...).

References:

- DCB macro-instructions and operands are described in detail in the publication IBM System/360 Operating System: Supervisor and Data Management Macro-Instructions.
- DCB macro-instructions and operands associated with the graphic access methods are described in the publications IBM System/360 Operating System: Graphic Programming Services for the IBM 2250 Display Unit, Form C27-6909, and Graphic Programming Services for IBM 2260 Display Station (Local Attachment), Form C27-6912.

Describing the Data Set Label (LABEL)

Data sets residing on magnetic tape volumes usually have data set labels. Direct-access volumes and volumes mounted through the automatic volume recognition feature must have labels conforming to standard label specifications. The LABEL parameter indicates the label type, the data set's relative position on tape, its retention period, and whether a password is required to read or write on it. To supplement this discussion of the LABEL parameter pictorially, turn to Appendix E and fold out Chart 3.

Magnetic tape volumes can contain volume labels and data set header and trailer labels that do not conform to the system standard label specifications. To create or retrieve a data set residing on such a tape volume, you must include the LABEL parameter. To specify the label type, code

```
LABEL=(,type)
```

in the operand field. Replace the word "type" with:

SL - if the data set has standard labels.
NL - if the data set has no labels.
NSL - if the data set has nonstandard labels.
SUL - if the data set has both standard and user labels.
BLP - to bypass label processing.

If you specify SUL, SL, or omit the label type (*in which case standard labels* are assumed), the operating system will

ensure that the correct volumes are mounted. If you specify NSL, your installation must have incorporated label processing routines into the operating system. If you specify NL, the data set must have no labels.

The feature that allows you to bypass label processing is a system generation option (OPTIONS=BYLABEL). If this option was not requested at system generation and you have coded BLP, the system assumes NL.

Note:

- When BLP is specified, you should ensure that the operator mounts the correct tape volume before processing it.

If the data set is not first in sequence on the reel, the LABEL parameter serves to position the tape properly through a data set sequence number. Code

```
LABEL=seq#
```

in the operand field. Replace the term "seq#" with the 1- to 4-digit sequence number assigned to the data set when it was created.

The sequence number describes the data set's position with respect to other data sets on the volume or group of volumes.

Notes:

- If you are retrieving a data set that resides on an unlabeled tape volume, you must give its sequence number with respect to other data sets on that single volume, beginning with 1.
- If 0 appears as the data set sequence number, the system assumes 1.

Both magnetic tape and direct-access data sets can be assigned a retention period and password protection when they are created. If you wish the data set to remain intact for some period of time, you can specify either the length of time in days or the exact date you want it to expire. Otherwise, a retention period of zero days is assumed. After expiration, the data set can be deleted, or opened for any type of output. To specify a retention period, code

```
LABEL=RETPD=nnnn
```

in the operand field. Replace the term "nnnn" with the number of days you want the data retained. To specify, instead, an expiration date, code

```
LABEL=EXPDT=yyddd
```

in the operand field. Replace the term "yyddd" with the 2-digit year number and 3-digit day number after which the data set can be considered expired.

If you wish the data set to be accessible only through the use of a password, code

```
LABEL=(,,PASSWORD)
```

in the operand field. The operating system assigns the data set security protection. To retrieve it, the operator must respond to a message by issuing the correct password.

Note:

- Subparameters in the LABEL parameter can be coded in various combinations. The terms seq#, type, and PASSWORD are all positional subparameters.

Specifying Data Set Status and Disposition (DISP)

The DISP parameter describes the status of a data set and indicates what is to be done with it after it is processed. You can omit this parameter if a data set is created and deleted during a single job step.

The first term in the DISP parameter reflects the data set's status with respect to the job step. If the data set existed before the job step, it can be used either as an input data set to be read or as a partially completed output data set. To specify the status of an existing data set used as input to a processing program, code

```
DISP=OLD
```

in the operand field of the DD statement. If the data set resides on a direct-access volume and is part of a job whose operations do not prevent simultaneous use of the data set by another job, code

```
DISP=SHR
```

in the operand field. This parameter has meaning only in a multiprogramming environment for existing data sets. If SHR is not coded in a multiprogramming environment, the data set is considered unusable by concurrently operating jobs. If SHR is coded in other than a multiprogramming environment, the system assumes the data set's status is OLD.

If a data set is sequentially organized, and is used for additional output by the processing program, code

```
DISP=MOD
```

in the operand field. When the data set is opened, the read/write mechanism is automatically positioned after the last record in the data set. If no volume information is available for the data set, the system assumes it does not yet exist and changes the MOD specification to NEW. (Volume information is considered available if it is coded in the DD statement, passed with the data set from a previous step, or contained in the catalog.)

A data set created in a job step is used by the processing program for output data. You can indicate a new status by coding

```
DISP=NEW
```

in the operand field, by coding DISP=MOD and including parameters usually required by new data sets, or by omitting a status specification altogether.

The second term in the DISP parameter tells how you want the data set handled by the job scheduler at the end of the job step that processes the data set. If the job scheduler can determine that the data set was not opened, the requested disposition is not performed.

Notes:

- If the job step is bypassed because of an error that occurs before the step is executed (e.g., an incorrect control statement is read, the system is not able to allocate units, etc.), requested dispositions are not performed. Subsequent job steps are also bypassed.
- If the job step is bypassed because of a return code test, requested dispositions are performed only for data sets that have been passed from a previous step.
- If the job step is bypassed because of an error that occurs during execution of the step (e.g., an incorrect volume label is encountered), requested dispositions are performed.

If you want the data set to assume the same status it had before the job step, you need not code a disposition. Existing data sets (OLD, MOD, and SHR) will continue to exist and newly created data sets (NEW) will be deleted. Special dispositions allow you to:

1. Uncatalog a data set.
2. Catalog a data set.
3. Delete an existing data set.
4. Keep a data set.
5. Pass a data set to a later job step.

These dispositions are discussed in the following numbered paragraphs.

1. To uncatalog an input data set, code the keyword parameter

   ```
   DISP=(OLD,UNCATLG)
   ```

   in the operand field. The catalog entry that points to the data set is removed from the index structure. If the data set resides on a direct-access volume, it remains tabulated in the volume table of contents.

2. To catalog a data set, code

```
DISP=(status,CATLG)
```

in the operand field. The term "status" reflects the data set's status, as discussed in earlier paragraphs. When you request cataloging, an index entry pointing to the data set is placed in the system catalog. The index structure required to catalog the data set must be defined before the cataloging operation can take place. DD statements in subsequent jobs can then refer to this data set simply by giving its cataloged name.

Notes:

- A cataloged data set whose status is MOD might be expanded to additional volumes during the job step. To update the catalog to reflect these additional volumes, code DISP=(MOD,CATLG).
- If the status of the data set is NEW, and you want to catalog it, you can omit the term NEW. However, you must indicate its absence with a comma, e.g., DISP=(,CATLG).

3. If you have no further need for a data set after its use and want to release its space, code

```
DISP=(status,DELETE)
```

in the operand field. The data set is automatically uncataloged if you have used the catalog to locate it. In addition, the system removes the volume table of contents entry associated with the data set, if it resides on a direct-access device.

Notes:

- If the status of the data set is NEW and you want to delete it after its use, i.e., the data set is temporary, you can omit the DISP parameter altogether.
- If you specify DISP=(SHR,DELETE), the system assumes OLD instead of SHR.

4. For data sets that are used in a later job but are not of sufficient importance to warrant their being cataloged, code

```
DISP=(status,KEEP)
```

in the operand field. The data set is kept intact until a DELETE request is encountered. If the volume containing the data set is demounted, the system advises the operator of the data set's KEEP disposition. If the data set resides on a direct-access volume, it remains tabulated in the volume table of contents.

Note:

- If the status of a data set is NEW and you want to keep it until a later time, you can omit the term NEW. However, you must indicate its absence with a comma, e.g., DISP=(,KEEP).

5. When a data set is used by two or more job steps in the same job, you can eliminate retrieval and disposal operations by passing it from step to step. Each step can use the data set one time. You do not indicate the final disposition of the data set until its last use in the job. To pass a data set to a succeeding step, code

```
DISP=(status,PASS)
```

in the operand field. Subsequent DD statements referring to the passed data set must identify it with the DSNAME parameter, must provide either no unit information, or unit information consistent with that in the original data set, and must issue another disposition. Between steps, the volume that contains the passed data set remains mounted; thus, you need not code RETAIN in the VOLUME parameter of a DD statement that specifies a disposition of PASS.

Notes:

- If the status of a data set is NEW and you want to pass it, you can omit the term NEW. However, you must indicate its absence with a comma, e.g., DISP=(,PASS).
- If the system finds it necessary to remove the volume containing a passed data set, it ensures through messages to the operator that the volume is remounted before its next use.

2. A data set can be placed in a specific position on a direct-access volume by requesting space in terms of a quantity and a track number. This allocation technique is recommended only for location-dependent data sets. To allocate tracks beginning at a specific address, code the keyword parameter

```
SPACE=(ABSTR,(quantity,address))
```

in the operand field of the DD statement. Replace the term "quantity" with the number of tracks you desire, and "address" with the relative track address of the beginning track. (The relative track address of the first track on the volume is 0. This track cannot be allocated.) If tracks you request have been allocated to another data set, the job is terminated.

Note:

- If the new data set is partitioned, you must indicate its directory size in the SPACE parameter by coding SPACE=(ABSTR,(quantity,address,directory)). Replace the term "directory" with the number of 256-byte blocks in the directory.

3. When a job step involves one or more data sets having corresponding records, you can minimize access arm movement by defining split cylinders. In the split-cylinder mode each data set is given a percentage of the tracks on every cylinder allocated.

   To split cylinders among two or more data sets, you must arrange the associated DD statements in sequence in the input stream. The first DD statement in the sequence specifies the portion of the space required by the first data set and the total amount of space required for all data sets. Each succeeding DD statement requests a portion of the total space.

   To request the total amount of space in units of cylinders, code the keyword parameter

```
SPLIT=(n,CYL,quantity)
```

   in the operand field of the first DD statement in the sequence. Replace the letter "n" with the number of tracks per cylinder you wish allocated to the first data set. Replace the term "quantity" with the total number of cylinders to be allocated for all the data sets. Each succeeding DD statement in the group must contain the parameter SPLIT=n, where n is the number of tracks per cylinder to be allotted to the associated data set.

   To request the total amount of space in units of blocks, code the keyword parameter

```
SPLIT=(%,blksize,quantity)
```

   in the operand field of the first DD statement in the sequence. Replace the character "%" with the percentage of tracks per cylinder you wish allocated to the first data set. Replace the terms "blksize" and "quantity" with the average length of the blocks in the data sets and the total number of blocks, respectively. The system computes the total number of cylinders from these figures. Each succeeding DD statement in the group must contain the parameter SPLIT=% in the operand field, where % is the percentage of tracks per cylinder to be allotted to the associated data set.

   Notes:

   - The SPLIT parameter cannot be used to allocate space for direct (BDAM) data sets or data sets residing on drum storage volumes.
   - The average block length cannot exceed 65,535 bytes.
   - If space is requested in units of blocks, and the blocks have keys, you must give the key length in the DCB parameter, i.e., KEYLEN=n.

   As in the SPACE parameter, you can ensure that increments of extra space will be automatically allocated by coding

```
SPLIT=(n,CYL,(quantity,increment))
```

   for increments in cylinders, or

```
SPLIT=(%,blksize,
              (quantity,increment))
```

   for increments in blocks. If any of the data sets in the group exceeds its allotted space, additional space is

allocated in the amount of the increment. (*If space* is requested in blocks, the system increases the increment to an integral number of cylinders.) The additional space applies only to the data set that exhausted its allotted space, and is not split with the other data sets in the group.

4. The fourth method of obtaining direct-access space is through the technique of suballocation. Suballocation allows you to place a number of data sets in contiguous order on a direct-access device. To use this technique, you simply request part of the space assigned to an earlier data set. Code

```
SUBALLOC=(units,(quantities),
                    stepname.ddname)
```

in the operand field of the DD statement. Replace the term "units" as in the SPACE parameter:

TRK - for a space request in tracks.
CYL - for a space request in cylinders.
average block length in bytes - for a space request in blocks.

The term "(quantities)" represents "(quantity,increment,directory)." The incremental quantity is optional and the number of directory blocks is required only when the DD statement defines a partitioned data set. If you indicate an incremental quantity, increments of space are allocated from available space on the volume, not from the space in the original data set. Replace the terms "stepname" and "ddname" with the names of the job step and the DD statement where an earlier data set is defined; the system suballocates the amount of space you request from this earlier data set.

Notes:

- Space obtained through suballocation must be contiguous, and cannot be further suballocated.
- The original data set must be used only for suballocation. Suballocated space is removed from the front of it.
- If the suballocation request refers to a DD statement in the same job step, you need not code the job step name, i.e., code the stepname, i.e., SUBALLOC=(units,(quantities),ddname).
- If the suballocation request refers to a DD statement in a cataloged procedure, you must include the name of the procedure step in which it appears, i.e., SUBALLOC=(units,(quantities),stepname.procstepname.ddname).

Optimizing Channel Usage (SEP and AFF)

A job step that requires several input and output operations might be performed more efficiently by balancing the channel requirements of its data sets. To obtain optimum channel usage, you can request that a data set be assigned a separate channel from the ones assigned to earlier data sets. A later DD statement can express the same separation requirements by requesting affinity.

To request channel separation from as many as eight other data sets in the job step, code the keyword parameter

```
SEP=(ddname,...,ddname)
```

in the operand field of the DD statement. Replace the terms "ddname" with the names of up to eight earlier DD statements in the job step.

To extend the channel separation facility, you can request affinity with an earlier data set that requested channel separation by coding the keyword parameter

```
AFF=ddname
```

in the operand field of a later DD statement. Replace the term "ddname" with the name of the earlier DD statement. The data set that requests affinity is also separated channelwise from those identified in the SEP parameter of the earlier statement. This feature eliminates your having to write identical SEP parameters more than once.

Note:

- Channel separation and affinity requests are ignored if the automatic volume recognition feature is used.

Creating Direct (BDAM) Data Sets

Direct (BDAM) data sets are created using the same subset of DD statement parameters as sequential data sets, with the exception of the SPLIT parameter. To distinguish a BDAM data set, you must code a DCB parameter containing one of the attributes DSORG=DA or DSORG=DAU. Valid DCB subparameters for BDAM data sets are listed in Appendix B.

Creating Partitioned (BPAM) Data Sets

Partitioned (BPAM) data sets are created using the same subset of DD statement parameters as sequential data sets, except that BPAM data sets cannot occupy a split cylinder. To distinguish a BPAM data set, you must code the number of directory blocks in the space allocation parameter. Valid DCB subparameters for BPAM data sets are listed in Appendix B.

Example 15 illustrates a DD statement for creating a BPAM data set with a 12-block directory.

```
//OUTPUT3  DD  DSNAME=X.Y.Z,VOLUME=REF=*.STEP1.OUTPUT,                X
//            DISP=(,CATLG),SPACE=(CYL,(20,,12))
```

Example 15. Creating and Cataloging a Partitioned Data Set, Using VOLUME=REF

Creating Indexed Sequential (BISAM and QISAM) Data Sets

Indexed sequential (ISAM) data sets are created using combinations of the DD statement parameters UNIT, DSNAME, VOLUME, LABEL, DISP, DCB, and SPACE. ISAM data sets occupy three areas of space: an index area that contains master and cylinder indexes: a prime area that contains the data records and track indexes; and an optional overflow area to hold additional records when the prime area is exhausted. Detailed information on creating and retrieving indexed sequential data sets is presented in Appendix D.

Creating Data Sets in the Output Stream

New data sets can be written on a system output device, much the same as messages. When using a sequential scheduler, you direct a data set to the output stream with the SYSOUT and DCB parameters.

SYSOUT: Required. Specify the standard output class, A.

DCB: Required only if complete data control block information has not been specified in the processing program.

Example 16 illustrates a DD statement for routing a data set through the output stream using a sequential scheduler.

```
//OUTPUT5  DD  SYSOUT=A
```

Example 16. Creating a SYSOUT Data Set (Sequential Scheduler)

When you are using a priority scheduler, data sets are not routed directly to a system output device. They are stored by the processing program on an intermediate direct-access device and later written on a system output device. In addition tc the SYSOUT and DCB parameters, DD statements defining a data set of this type can also contain UNIT and SPACE parameters. All other parameters must be absent.

SYSOUT: Required. Specify the output class through which you want the data set routed. Output classes are identified by a single alphameric character.

DCB: Required only if complete data control block information has nct been specified in the processing program. Data control block information is used when the data set is written on an intermediate direct-access volume and read by the cutput writer. However, the output writer's own DCB attributes are used when the data set is written on the system output device. Valid DCB parameters are listed in Appendix B.

UNIT: Optional. Assign an intermediate direct-access device. A default device is assigned if you omit this parameter.

SPACE: Optional. Estimate the amount of direct-access space required. A default estimate is assumed if you omit this parameter.

Example 17 illustrates a DD statement for routing a data set through an output stream using a priority scheduler.

```
//OUTPUT5  DD  SYSOUT=F,UNIT=2311,SPACE=(TRK,(10,2))
```

Example 17. Creating a SYSOUT Data Set (Priority Scheduler)

Optimizing Channel Usage

Optimum channel usage can be obtained in some job steps by requesting that certain data sets be assigned separate channels from others. This facility should be used only when a significant saving of time can be realized by using separate channels. It considerably restricts device allocation and may result in unnecessary dismounting of volumes.

The system treats channel separation and affinity requests on an "if available" basis, that is, they will be recognized only if enough channels are available. If one request cannot be satisfied, all such requests in the step may be ignored. In addition, requests are ignored if the automatic volume recognition feature is used. Example 34 illustrates a job step containing channel requests.

```
//STEP1    EXEC  PGM=CONVERT
//INPUT1   DD    DSNAME=A.B.C,DISP=OLD
//INPUT2   DD    DSNAME=X.Y.Z,DISP=OLD
//BUF      DD    UNIT=2400,SEP=(INPUT1,INPUT2)
//OUTPUT   DD    DSNAME=ALPHA,UNIT=TAPE,DISP=(,KEEP),AFF=BUF
```

Example 34. Requesting Channel Separation and Affinity

If enough channels are available, the temporary data sets defined by the DD statements BUF and OUTPUT are assigned a channel *other than* the ones used by data sets A.B.C and X.Y.Z. Note that the data sets defined by BUF and OUTPUT may or may not use the same channel.

USING CATALOGED PROCEDURES

Applications that require many control statements and are used on a regular basis can be considerably simplified through the use of cataloged procedures. A cataloged procedure is a set of job control statements that has been placed in a partitioned data set known as the procedure library. (The procedure library is a system data set named SYS1.PROCLIB.) You retrieve a catalcged procedure from the library by using its member name in an EXEC statement in the input stream. Other control statements in the input stream can be used to temporarily override or add to the control statements in a procedure.

Establishing Cataloged Procedures

Cataloged procedures contain one or more EXEC statements, each followed by associated DD statements. Each EXEC statement and its associated DD statements represent a procedure step. EXEC statements identify programs to be executed. Cataloged procedures cannot contain:

- EXEC statements referring to other cataloged procedures.
- JOB, command, delimiter, or null statements.
- DD statements with the ddname JOBLIB.
- DD statements with * or DATA coded in the operand field.

You add cataloged procedures to the procedure library by using the IEBUPDTE utility program. You can also use this program to permanently modify existing procedures. A description of this program and examples of its use are contained in the publication IBM System/360 Operating System: Utilities.

Overriding EXEC Statements in Cataloged Procedures

To override or add to the parameters on an EXEC statement in a cataloged procedure, you must include the procedure step name as part of the keywords on the EXEC statement that calls the procedure, i.e., TIME.procstepname. Each such keyword can appear as many times as there are steps in the procedure. However, you must code all overriding parameters for one procedure step before those of the next step.

If you wish to override all EXEC statement parameters in a cataloged procedure with a single set of parameter values, code the overriding keywords by themselves. Overriding parameters of this type modify as follows:

- PARM -- applies to the first step in the procedure and nullifies all other PARM parameters.
- COND, ACCT -- apply to every step in the procedure.
- TIME, REGION -- override all TIME and REGION parameters and apply to the entire procedure.

Overriding and Adding DD Statements

To override the DD statement parameters in a cataloged procedure, or to add data sets to the procedure, you must include some DD statements in the input stream. These DD statements must have a name field of the form procstepname.ddname, e.g., STEP1.OUTPUT. The terms "procstepname" and "ddname" identify the procedure step and DD statement that you are overriding. If you are adding a data set to the procedure, the term "ddname" must be different from other ddnames in the procedure step.

There are a few rules you should keep in mind when overriding or adding to a cataloged procedure step:

1. Overriding DD statements must be in the same order in the input stream as the corresponding DD statements in the cataloged procedure.

2. DD statements to be added must follow overriding DD statements for the step.

3. If you are using a sequential scheduler and one of the overriding or additional DD statements has an * in its operand field, it must be last.

To override a parameter in a procedure DD statement, either (1) recode the entire parameter on the overriding statement, modifying it as you wish, or (2) code a suitable replacement for the parameter, such as SPLIT for SPACE. The DCB parameter is an exception to these rules; you need only recode the DCB attributes you wish to modify. Parameters that you do not wish to override need not be recoded.

To nullify a keyword parameter, code the keyword followed by an equal sign in the overriding DD statement, e.g., UNIT=. To nullify a DUMMY parameter, code the DSNAME parameter in the overriding DD statement, but do not use the data set name NULLFILE.

Example 35 shows an existing cataloged procedure named ANALYSIS. This procedure is used and modified by the statements in Example 36 to produce the temporary result as shown in Example 37.

```
//LOOKUP   EXEC  PGM=SEARCH
//IN1      DD    DSNAME=A.B.C,DISP=CLD
//OUT1     DD    UNIT=2311,SPACE=(TRK,(10,2)),DISP=(,PASS)
//REDUCE   EXEC  PGM=TRUNCATE
//IN2      DD    DSNAME=*.LOOKUP.OUT1,DISP=(OLD,DELETE)
//WORK     DD    UNIT=TAPE
//OUT2     DD    UNIT=2311,SPACE=(TRK,(5,1)),DISP=(,PASS)
//DISPLAY  EXEC  PGM=PRINT
//IN3      DD    DSNAME=*.REDUCE.OUT2,DISP=(OLD,DELETE)
//OUT3     DD    SYSOUT=A
```

Example 35. Modifying a Cataloged Procedure -- The Procedure

The procedure in Example 35 comprises three steps, each of which receives input data from the catalog or the previous step, processes the data, and places it in an output data set. The second step makes use of a temporary work file on tape.

```
//STEP1         EXEC   ANALYSIS
//LOOKUP.OUT1   DD     UNIT=2400,DISP=
//REDUCE.WORK   DD     UNIT=180
//REDUCE.XTRA   DD     UNIT=181
//DISPLAY.IN3   DD     DISP=(OLD,KEEP)
```

Example 36. Modifying a Cataloged Procedure -- The Input Stream

The job step in Example 36 uses the procedure and modifies it by:

- Changing a disk to a tape and nullifying a disposition (implying a disposition of DELETE).
- Specifying a specific unit for the work file.
- Adding an additional work file.
- Changing a disposition.

As a result of these modifications, the procedure temporarily appears as shown in Example 37. As a result of the unit change in the DD statement OUT1, the SPACE parameter is ignored.

```
//LOOKUP    EXEC  PGM=SEARCH
//IN1       DD    DSNAME=A.B.C,DISP=CLD
//OUT1      DD    UNIT=2400,SPACE=(TRK,(10,2))
//REDUCE    EXEC  PGM=TRUNCATE
//IN2       DD    DSNAME=*.LOOKUP.OUT1,DISP=(OLD,PASS)
//WORK      DD    UNIT=180
//XTRA      DD    UNIT=181
//OUT2      DD    UNIT=2311,SPACE=(TRK,(5,1)),DISP=(,PASS)
//DISPLAY   EXEC  PGM=PRINT
//IN3       DD    DSNAME=*.REDUCE.OUT2,DISP=(OLD,KEEP)
//OUT3      DD    SYSOUT=A
```

Example 37. Modifying a Cataloged Procedure -- The Result

Using the DDNAME Parameter

The DDNAME parameter, used primarily in cataloged procedures and job steps that call procedures, allows you to define a dummy data set that can, at a later point in the same job step, assume real data set characteristics. This facility is primarily used when a cataloged procedure receives data from an input stream.

To use this facility, code DDNAME=ddname in the operand field of a DD statement. This DD statement represents a data set whose definition is postponed until a DD statement with a matching ddname in its name field is encountered. At this point, all of the data set characteristics defined in the matching statement are applied to the original statement.

There are a few rules you should keep in mind as you use the DDNAME parameter:

1. You cannot make a backward reference (i.e., *.ddname) to a DD statement whose ddname matches a previous DDNAME parameter. Also you cannot place unnamed DD statements after such a statement (i.e., concatenate).

The UNIT parameter of the DD statement can identify an input or output unit by its actual address, its type number, or its group name. Type numbers, automatically established at system generation, correspond to units configured into your system. Type numbers and corresponding units are listed here for your convenience.

Tape Units

Unit Type	Unit
2400	any 2400 Nine-Track Magnetic Tape Drive
2400-1	any 2400 Magnetic Tape Drive with Seven-Track Compatibility
2400-2	2400 Magnetic Tape Drive with Seven-Track Compatibility and Data Conversion
2400-4,-5,-6	any 2400 Nine-Track Magnetic Tape Drive with a density of 1600 bytes per inch. Optional feature allows for a density of 800 bytes per inch. Model number denotes rate of data transmission.

Direct Access Units

Unit Type	Unit
2301	2301 Drum Storage Unit
2302	2302 Disk Storage Drive
2303	2303 Drum Storage Unit
2311	any 2311 Disk Storage Drive
2314	2314 Storage Facility

Unit Record Equipment

Unit Type	Unit
1052	1052 Printer-Keyboard
1403	1403 Printer or 1404 Printer (continuous form only)
1442	1442 Card Read Punch
1443	any 1443 Printer
2501	2501 Card Reader
2520	2520 Card Read Punch
2540	2540 Card Read Punch (read feed)
2540-2	2540 Card Read Punch (punch feed)
2671	2671 Paper Tape Reader

Graphic Units

1053	1053 Printer
2250	2250 Display Unit, Model 1
2250-2	2250 Display Unit, Model 2
2250-3	2250 Display Unit, Model 3
2260	2260 Display Station (local attachment)
2280	2280 Film Recorder
2282	2282 Film Recorder/Scanner

APPENDIX B: DCB SUBPARAMETERS

The data control block associated with a data set is filled by a number of sources, one of which is the DD statement. The DCB parameter supplies missing or overriding attributes in the form of a list of subparameters. The glossary below lists the keywords that you can code in the DCB parameter, their definitions, and the values they may assume.

Table 3 supplies valid keywords and values that you can use with the indexed sequential, partitioned, direct, and sequential access methods. Subparameters that apply to the graphics access method (GNCP,GDSORG) and the teleprocessing access methods (BUFRQ,CPRI,INTVL,SOWA) do not appear in the table. Further information on DCB subparameters appears in the publication IBM System/360 Operating System: Supervisor and Data Management Macro-Instructions.

Glossary of DCB Subparameters

BFALN Fullword (F) or doubleword (D) boundary alignment of each buffer.

BFTEK Type of buffering (simple or exchange) to be supplied by the control program (S or E).

BLKSIZE Maximum block size in bytes (a number).

BUFL Length, in bytes, of each buffer to be obtained for a buffer pool (a number).

BUFNO Number of buffers to be assigned to the data control block.

BUFRQ Number of buffers to be read in advance from the direct-access device queue. (For use with teleprocessing access methods.)

CODE Paper tape code in which the data is punched.

- I - IBM BCD perforated tape and transmission code (8 tracks)
- F - Friden (7 tracks)
- B - Burroughs (7 tracks)
- C - National Cash Register (8 tracks)
- A - ASCII (8 tracks)
- T - Teletype (5 tracks)
- N - No conversion

CPRI Relative priority to be given to sending and receiving operations. (For use with teleprocessing access methods.)

CYLOFL Number of tracks to be reserved on each cylinder to hold records that overflow from other tracks on that cylinder.

DEN Tape recording density.

- 0 - 200 bits/inch (7-track only)
- 1 - 556 bits/inch (7-track only)
- 2 - 800 bits/inch

For 7-track tapes, all information on the reel must be written in the same density (i.e., labels, data, tapemarks). Do not specify DEN for a SYSOUT data set.

DSORG Organization of the data set.

- PS - Physical sequential
- PSU - Physical sequential unmovable
- PO - Partitioned organization
- POU - Partitioned organization unmovable
- IS - Indexed sequential
- ISU - Indexed sequential unmovable
- DA - Direct-access
- DAU - Direct-access unmovable

EROPT Option to be executed if an error occurs.

- ACC - Accept
- SKP - Skip
- ABE - Abnormal end of task

GDSORG Organization of a graphic data set. (For use with the graphics access method.)

GNCP Maximum number of input/output macro-instructions that will be issued before a WAIT macro-instruction. (For use with the graphics access method.)